Example - Newton-Raphson Method

We now consider the following example:

$$\text{minimize } f(x) = -x^3 + \frac{3}{4}x^4$$

Since $f'(x) = -3x^2 + 3x^3$ and $f''(x) = -6x + 9x^2$ we form the following iteration:

$$x^{n+1} = x^n - \frac{3(x^n)^3 - 3(x^n)^2}{9(x^n)^2 - 6x^n} = x^n - \frac{(x^n)^2 - x^n}{3x^n - 2} = \frac{2(x^n)^2 - x^n}{3x^n - 2}.$$

We can now try the iteration for different initial guesses.

initial guess: $x^{(0)} = 0$

$$(x^{(1)} = 0) \Rightarrow \text{stationary point}$$

initial guess: $x^{(0)} = 0.5$

$$(x^{(1)} = 0) \Rightarrow \text{stationary point}$$

initial guess: $x^{(0)} = .9$

$$x^{(1)} = 1.029$$

$$x^{(2)} = 1.0015$$

$$(x^{(3)} = 1.0000045) \Rightarrow 1.0 = \text{stationary point}$$

initial guess: $x^{(0)} = 1.0$

$$x^{(1)} = 1.0$$

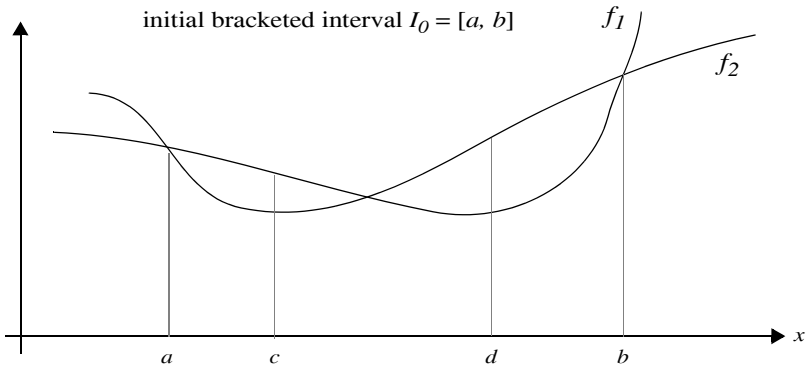
We can now evaluate *f(x)* at the two stationary points we've found:

$$f(0) = 0, f(1) = -1/4$$

and since $f''(1) = 3 > 0 \Rightarrow \text{local minimum}$ therefore $x^* = 1 \Rightarrow \text{local minimum}$.

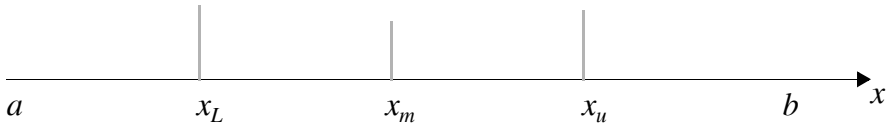
In fact you can convince yourself that it is the global minimum. (Note that the global maximum is at $x^* = \pm\infty$.)

Interval Reduction Using Function Samples Only



1) consider $f_1 : f_1(d) < f_1(c) \Rightarrow I_1 = [c,b]$,

2) consider $f_2 : f_2(d) > f_2(c) \Rightarrow I_1 = [a,d]$.



Algorithm: Interval Halving Search

1. input *a, b*, Δmin // input the initial interval
2. set: $\Delta = b - a$, $x_m = a + \Delta/2$, $f_m = f(x_m)$
3. while $\Delta > \Delta min$ repeat
4. set: $x_L = a + \Delta/4$, $f_L = f(x_L)$
5. $x_u = b - \Delta/4$, $f_u = f(x_u)$
6. if $f_L < f_m$ then
7. set $b = x_m, x_m = x_L, f_m = f_L, \Delta = b - a$
8. else if $f_u < f_m$
9. set $a = x_m, x_m = x_u, f_m = f_u, \Delta = b - a$
10. else
11. set $a = x_L, b = x_u, \Delta = b - a$
12. end if
13. end if
14. end while
15. output [*a, b*] // final interval size is less than Δmin

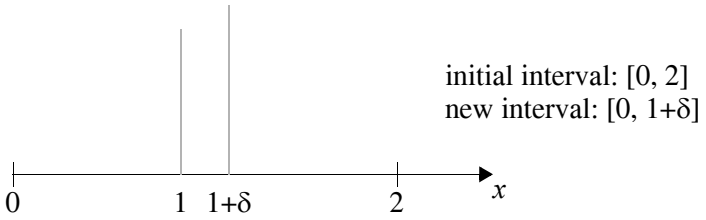
After $2n+1$ function evaluation the interval is reduced by

$$I_{2n+1} = I_0(1/2)^n, \; n = 1, 2, 3\ldots$$

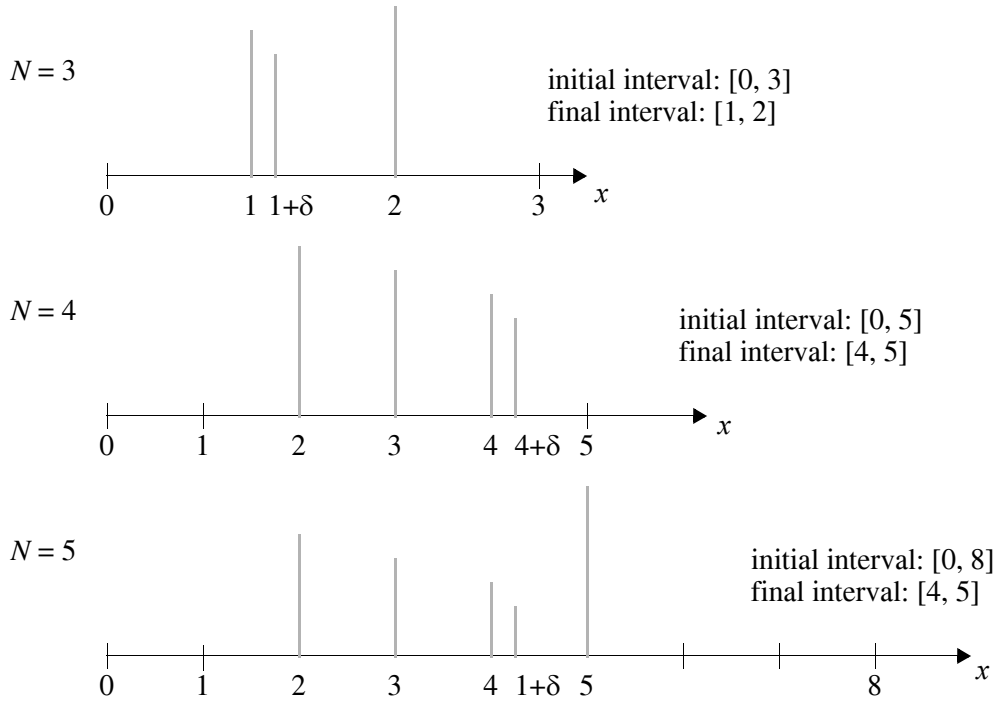
where I_o is the size of the initial interval.

Fibonacci Search Technique

Question How should we pick N successive points in an interval to perform function evaluations such that the final reduced interval is the smallest possible irrespective of the properties of $f(x)$?



Best location for two function evaluations

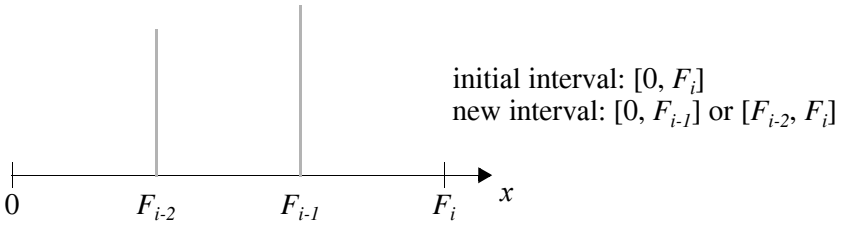


Examples of optimal sampling locations

Fibonacci sequence of numbers:

$$F_0 = F_1 = 1, F_i = F_{i-1} + F_{i-2}, i = 2(1)\infty.$$

Given an interval of proportion F_i and choosing two sample points at F_{i-2}, F_{i-1}.



Fibonacci type sampling

The *new interval* will be either

$$[0,F_{i-1}] \text{ or } [F_{i-2},F_i]$$

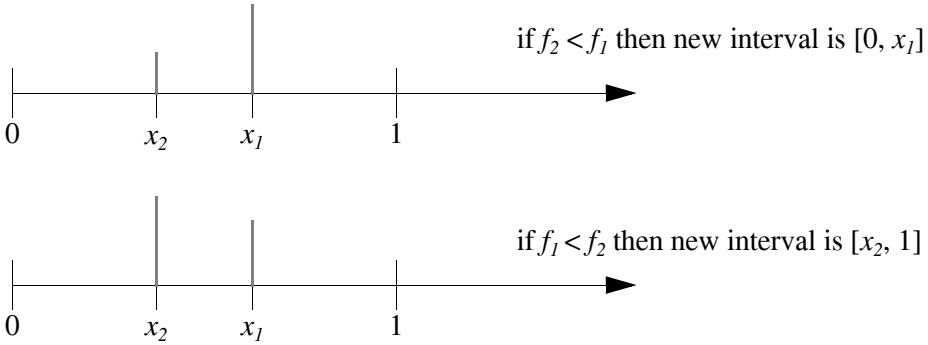
the first has size F_{i-1}, the second has size $F_i - F_{i-2} = F_{i-1}$ and therefore they are the same size. The next point is chosen as either F_{i-3} or $F_i - F_{i-3}$.

The interval reduction is equal to $I_N/I_0 = 1/F_N$ after *N* function evaluations. It has been shown that this is the best possible reduction for a given number of function evaluations. At each step the reduction is equal to F_i/F_{i+1}. The drawback of this procedure is that you must start with F_N and work backwards down the sequence.

Golden Section Search

Given a normalized interval $[0,1]$ where should two points x_1 and x_2 be chosen such that:

1) size of reduced interval is independent of function.
2) only one function evaluation per interval reduction.



Golden section interval reduction, initial interval is [0, 1].

Therefore by condition 1 above

$$x_1 - 0 = 1 - x_2 \text{ or } x_2 = 1 - x_1$$

and by condition 2

$$\frac{x_1 - x_2}{x_1} = \frac{1 - x_1}{1}$$

$$x_1 - x_2 = x_1 - x_1^2 \Rightarrow x_1^2 = x_2$$

from

$$x_1^2 + x_1 - 1 = 0 \Rightarrow x_1 = \tau = .61803$$

$$x_2 = 1 - x_1 = .38197 = \theta = 1 - \tau = \tau^2$$

where we call τ the golden section and θ the golden mean.

After each function evaluation and comparison θ of the interval is eliminated or τ of the original interval remains. Thus after *N* function evaluations an interval of original size *L* will be reduced to a size

$$L\tau^{N-1}$$

Algorithm: Golden Section search

1. input a_1, b_1, tol
2. set $c_1 = a_1 + (1 - \tau)(b_1 - a_1), F_c = F(c_1)$
3. $d_1 = b_1 - (1 - \tau)(b_1 - a_1), F_d = F(d_1)$
4. for $K = 1, 2, \ldots$ repeat
5. if $F_c < F_d$ then
6. set $a_{k+1} = a_k, b_{k+1} = d_k, d_{k+1} = c_k$
7. $c_{k+1} = a_{k+1} + (1 - \tau)(b_{k+1} - a_{k+1})$
8. $F_d = F_c, F_c = F(c_{k+1})$
9. else
10. set $a_{k+1} = c_k, b_{k+1} = b_k, c_{k+1} = d_k$
11. $d_{k+1} = b_{k+1} - (1 - \tau)(b_{k+1} - a_{k+1})$
12. $F_c = F_d, F_d = F(d_{k+1})$
13. end
14. end until $b_{k+1} - a_{k+1} < tol$

Notes: iterate until $\|\text{interval}\| < tol$

Polynomial Interpolation Methods

In an interval of uncertainty the function is approximated by a polynomial and the minimum of the polynomial is used to predict the minimum of the function.

Quadratic Interpolation

If $F(x) = px^2 + qx + r$ is the interpolating quadratic we need to find the coefficients *p*, *q*, and *r* given the end points of the interval $[a,b]$ and *c* such that $a < c < b$ with $F_a = F(a)$ $F_b = F(b)$, $F_c = F(c)$. The coefficients satisfy the matrix equation

$$\begin{bmatrix} a^2 & a & 1 \\ b^2 & b & 1 \\ c^2 & c & 1 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} F_a \\ F_b \\ F_c \end{bmatrix}$$

which can be easily solved analytically. Once we have the coefficients we need to find the minimum of *F(x)* and thus

$$\frac{d}{dx}F(x) = g(x^*) = 2px^* + q = 0$$

$$x^* = -\frac{q}{2p}$$

and therefore we only need the *q* and *p* coefficients. These can be solved using Cramer's rule:

$$p = \frac{1}{\Delta}\begin{vmatrix} F_a & a & 1 \\ F_b & b & 1 \\ F_c & c & 1 \end{vmatrix} = \frac{F_a(b-c) + F_b(c-a) + F_c(a-b)}{\Delta}$$

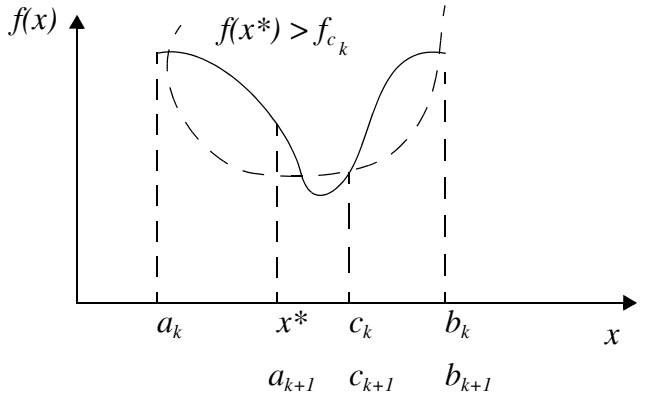
$$q = \frac{1}{\Delta}\begin{vmatrix} a^2 & F_a & 1 \\ b^2 & F_b & 1 \\ c^2 & F_c & 1 \end{vmatrix} = -\frac{F_a(b^2-c^2) + F_b(a^2-c^2) - F_c(a^2-b^2)}{\Delta}$$

and therefore

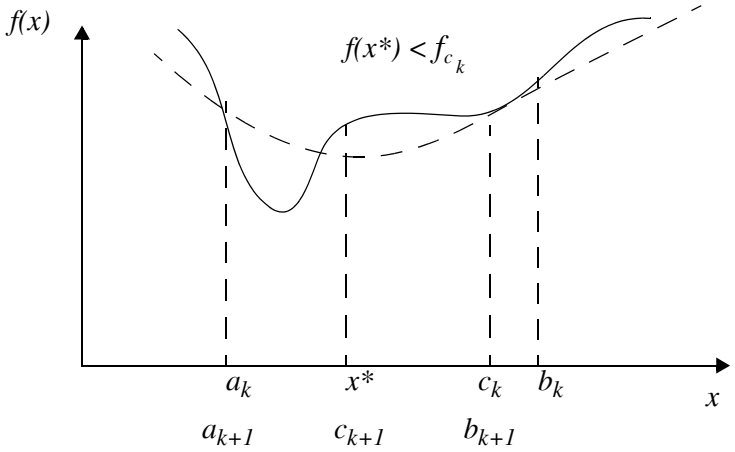
$$x^* = -\frac{q}{2p} = \frac{1}{2}\frac{F_a(b^2-c^2) + F_b(c^2-a^2) + F_c(a^2-b^2)}{F_a(b-c) + F_b(c-a) + F_c(a-b)}$$

Notice that x^* must lie in $[a,b]$ if $F_c < F_a$ and $F_c < F_b \Leftarrow$ bracketing interval. Also note that $f''(x^*) = 2p(x^*)$ and therefore if $p(x^*) > 0 \Rightarrow$ local minimum of quadratic whereas if $p(x^*) < 0 \Rightarrow$ local maximum.

We now evaluate $f(x^*)$ and compare it to $f(c)$ in order to reduce the interval of uncertainty.



$f(x^*) > f_{c_k} \Rightarrow$ new interval is $[a_{k+1}, b_{k+1}]$ and new c value is c_{k+1}



$f(x^*) < f_{c_k} \Rightarrow$ new interval is $[a_{k+1}, b_{k+1}]$ and new c value is c_{k+1}

Algorithm: Quadratic interpolation search

1. input $a_1, b_1, c_1, xtol, ftol$
2. set $F_a = F(a_1), F_b = F(b_1), F_c = F(c_1)$
3. for $k = 1,2, \ldots$ repeat
4. set $x^* = \frac{1}{2}\frac{F_a(b_k^2 - c_k^2) + F_b(c_k^2 - a_k^2) + F_c(a_k^2 - b_k^2)}{F_a(b_k - c_k) + F_b(c_k - a_k) + F_c(a_k - b_k)}$
5. $F_x = F(x^*)$
6. if $x^* > c_k$ and $F_x < F_c$ then
7. set $a_{k+1} = c_k, b_{k+1} = b_k, c_{k+1} = x^*$
8. $F_a = F_c, F_c = F_x$
9. else if $x^* > c_k$ and $F_x > F_c$ then
10. set $a_{k+1} = a_k, b_{k+1} = x^*, c_{k+1} = c_k$
11. $F_b = F_x$
12. else if $x^* < c_k$ and $F_x > F_c$ then
13. set $a_{k+1} = x^*, b_{k+1} = b_k, c_{k+1} = c_k$
14. $F_a = F_x$
15. else
16. set $a_{k+1} = a_k, b_{k+1} = c_k, c_{k+1} = x^*$
17. $F_b = F_c, F_c = F_x$
18. end
19. end until $(b_{k+1} - a_{k+1}) < xtol$ or $(F(c_k) - F(c_{k+1}))/F(c_k) < ftol$

Notes: stop when interval reduced to *xtol* or when relative change in function value < *ftol*

Bounding Phase

Question: how do we determine this initial interval?

In general we perform a coarse search to *bound* or *bracket* the minimum x^*. This is also called *interval location*. We will discuss two types of interval location:

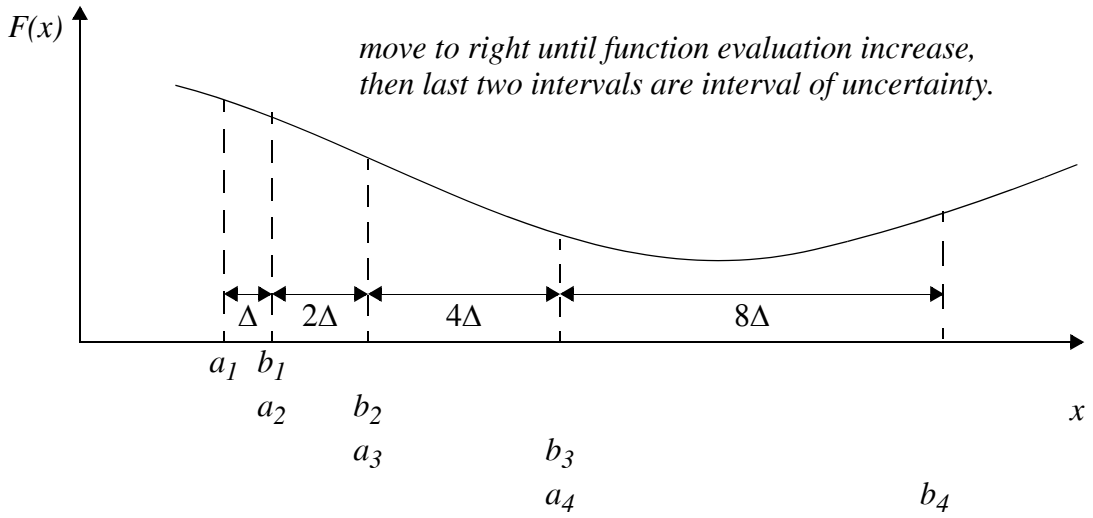
1) function comparison (Swann's Bracketing Method); and

2) polynomial extrapolation (Powell's Method).

Both of these methods are heuristic in nature but are about the best we can provide in terms of an algorithm. We start with a description of Swann's bracketing method.

Function comparison: Swann's Bracketing Method

- Given an initial step length Δ and starting point a_1 we check the point a distance Δ away on the right side of a_1.
- If $f(a_1 + \Delta) < f(a_1)$ then we will move to the right.
- If $f(a_1 + \Delta) > f(a_1)$ then we will move to the left (lets say we move to the right).
- We now magnify the step size by a magnification factor, say 2, and evaluate the function at the point 2Δ away from the latest point.
- As soon as any new function evaluation shows an increase we can say that the last two intervals provide a bound or interval of uncertainty for our minimum.



Example of Swann's bracketing method, magnification factor is 2.0.

Algorithm: Swann's Bracketing Method - based on a heuristic expanding pattern.

1. input: x_0, Δ // initial point and step size
2. set: $x_l = x_0 - |\Delta|$ // lower test point.
3. $x_u = x_0 + |\Delta|$ // upper test point.
4. $f_l = f(x_l)$ // lower function value
5. $f_u = f(x_u)$ // upper function value
6. $f_0 = f(x_0)$ // central function value
7. $i = 1$ // expanding exponent
8. if $f_l \geq f_0 \geq f_u$ // **case 1**: move to the right
9. *loop-up*: $i = i + 1$
10. set: $x_l = x_0, x_0 = x_u, f_l = f_0, f_0 = f_u$
11. $x_u = x_u + 2^i|\Delta|$ // shift up in x by a magnified amount
12. $f_u = f(x_u)$
13. if $f_u < f_0$ go to *loop-up*
14. else output (x_l, x_u)
15. if $f_l \leq f_0 \leq f_u$ // **case 2**: move to the left
16. *loop-down*: $i = i + 1$
17. set: $x_u = x_0, x_0 = x_l, f_u = f_0, f_0 = f_l$
18. $x_l = x_l - 2^i|\Delta|$ // shift down in x by a magnified amount
19. $f_l = f(x_l)$
20. if $f_l < f_0$ go to *loop-down*
21. else output (x_l, x_u)
22. if $f_l \geq f_0 \leq f_u$ // **case 3**: initial interval is bracket
23. output (x_l, x_u)
24. if $f_l \leq f_0 \geq f_u$ *error*: non-unimodal function

Polynomial Extrapolation: Powell's Method

- we use polynomial extrapolate from the initial starting point.
- Given a starting point a, and two more points Δ apart we *extrapolate* a polynomial through them (we will use a quadratic function).
- Now in this case the fitted quadratic may have either a maximum or minimum and this may be determined by evaluating the second derivative.

- As before if the polynomial is represented as

$$F(x) = px^2 + qx + r$$

then the second derivative is given as

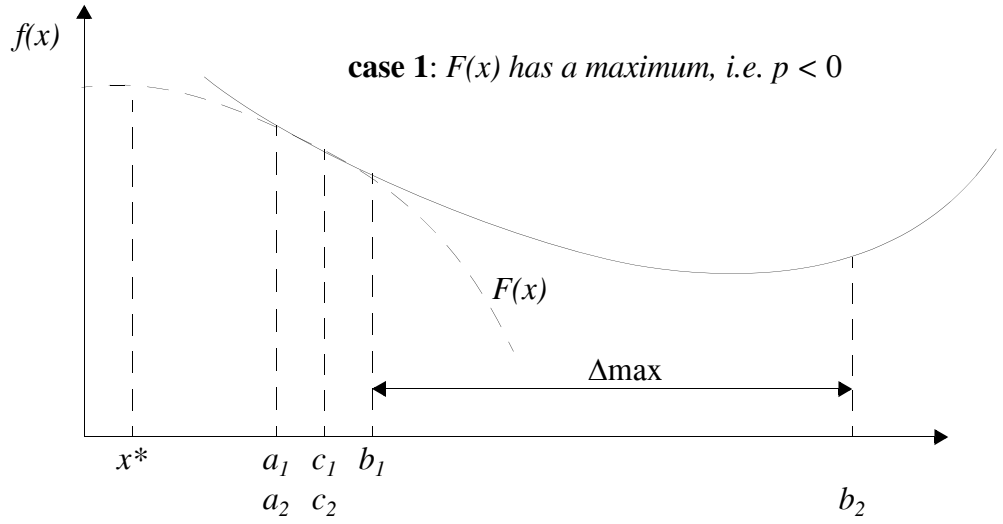
$$F''(x) = G(x) = 2p$$

and in terms of the three function values F_a, F_b, F_c, evaluated at $x = a$, b, and c we can evaluate p as

$$p = \frac{(c-b)F_a + (a-c)F_b + (b-a)F_c}{(b-c)(c-a)(a-b)}.$$

Now how do we use this to find an interval?

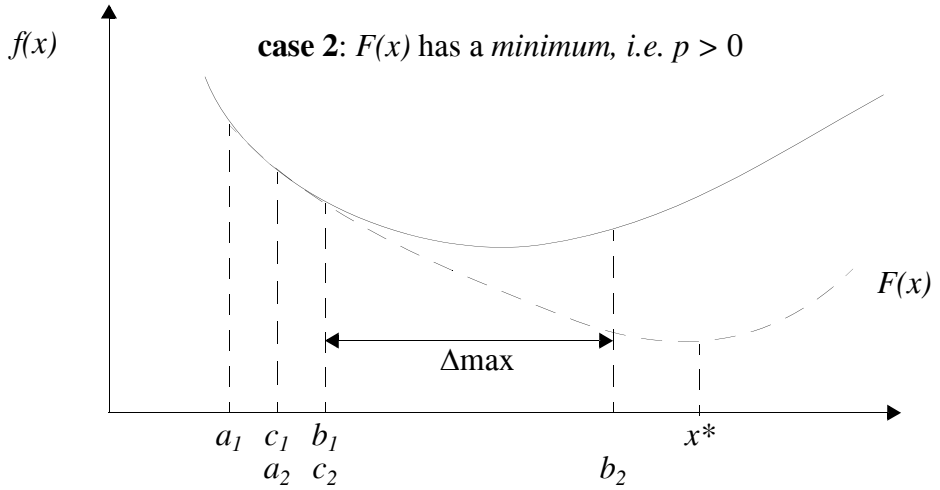
- Starting from three points, a_1, c_1, and b_1, a Δ apart we fit a quadratic to these points, say $F(x)$.
- If we find that this quadratic has a maximum, that is $p < 0$, then the next point we take is Δmax away from the smallest function value. (See figure below) where for sake of an example we are assuming that we move to the right.



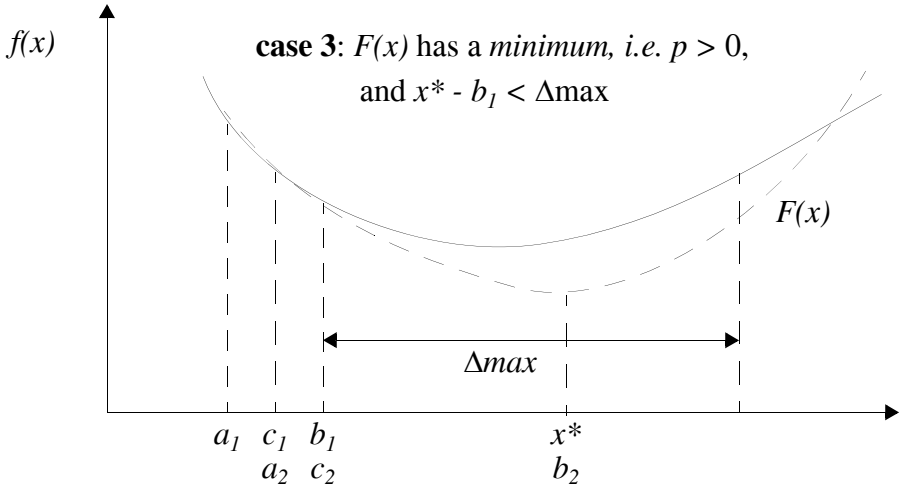
Extrapolated polynomial has a maximum therefore use the Δmax.

- We discard the point with smallest function value and take new point Δmax away from b_1. Discarding the lowest function value gives us more of a chance of fitting a polynomial with a minimum in the next iteration of the algorithm.

- If the polynomial does have a minimum then we either use the minimum of the polynomial as shown in case 3 or we again use Δmax if the minimum is farther than Δmax from b_1 as shown in case 2.



Extrapolated polynomial has a minimum but it is too far away so use Δmax.



Extrapolated polynomial has a minimum and it is closer than Δmax from b_1.

Algorithm: Interval location by Powell's Method

1. input a_1, Δ, $\Delta\max$
2. set $c_1 = a_1 + \Delta$, $F_a = F(a_1)$, $F_c = F(c_1)$
3. if $Fa > Fc$ then
4. set $b_1 = a_1 + 2\Delta$, $F_b = F(b_1)$
5. *forward* = true
6. else
7. set $b_1 = c_1$, $c_1 = a_1$, $a_1 = a_1 - \Delta$
8. $F_b = F_c$, $F_c = F_a$, $F_a = F(a_1)$
9. *forward* = false

10. end
11. for $K = 1, 2, \ldots$ repeat
12. set $p = \dfrac{(c_K - b_K)F_a + (a_K - c_K)F_b + (b_K - a_K)F_c}{(b_k - c_K)(c_K - a_K)(a_K - b_K)}$
13. if $p > 0$ then
14. set $x^* = \dfrac{1}{2}\dfrac{(b_K^2 - c_K^2)F_a + (c_k^2 - a_K^2)F_b + (a_K^2 - b_K^2)F_c}{(b_K - c_K)F_a + (c_K - a_K)F_b + (a_K - b_K)F_c}$
15. end if
16. if *forward* then // moving forward
17. if $p \le 0$ then // quadratic has a maximum
18. set $a_{K+1} = a_K, b_{K+1} = b_K + \Delta\max$
19. $c_{K+1} = c_K, F_b = F(b_{K+1})$
20. else
21. if $x^* - b_K > \Delta\max$ then // quadratic minimum is too far
22. set $b_{K+1} = b_K + \Delta\ \max$
23. else // quadratic minimum is O.K.
24. set $b_{K+1} = x^*$
25. end
26. set $a_{K+1} = c_K, c_{K+1} = b_K$
27. $F_a = F_c, F_c = F_b, F_b = F(b_{K+1})$
28. end
29. else // moving backward (*forward* = false)
30. if $p \le 0$ then // quadratic has a maximum
31. set $a_{K+1} = a_K - \Delta\max, b_{K+1} = b_K$
32. $c_{K+1} = c_K, F_a = F(a_{K+1})$
33. else
34. if $a_K - x^* > \Delta\max$ then // quadratic minimum is too far
35. set $a_{K+1} = a_K - \Delta\max$
36. else // quadratic minimum is O.K.
37. set $a_{K+1} = x^*$
38. end
39. set $b_{K+1} = c_K, c_{K+1} = a_K$
40. $F_b = F_c, F_c = F_a, F_a = F(a_{K+1})$
41. end // if $p \le 0$
42. end
43. end until $F_c < F_a$ and $F_c < F_b$